Final

Remote Access Multi-mission Processing and Analysis Ground Environment (RAMPAGE)[1]

Young H. Lee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-1326
Young.H.Lee@jpl.nasa.gov

Ted E. Specht
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-9645
Ted.E.Specht@jpl.nasa.gov

Abstract—
At Jet Propulsion Laboratory (JPL), a goal of providing easy and simple data access to the mission engineering data using web-based standards to a wide variety of users is now possible by the RAMPAGE development. Because RAMPAGE provides platform independence by utilizing a freely available web browser instead of a special purpose workstation, a myriad of users are able to participate in multiple missions from any location via a virtual workstation environment. The first stage of RAMPAGE prototype was envisioned and developed while supporting the Galileo project and then to be followed by Deep Space 1 DS1) and Cassini project. The plan is to deploy the production system of RAMPAGE to support Mars 01 orbiter, Lander and Rover project, which is to be launched around March and April 2001. Discussions are also ongoing to deploy RAMPAGE to the Space InfraRed Telescope Facility (SIRTF) project which is to be launched December 2001.

TABLE OF CONTENTS

1.0 INTRODUCTION

In today's increasingly complicated missions, spacecraft, and instrument engineers, science investigators, and flight controllers depend upon reliable displays and analysis tools to utilize engineering data. There are various forms of engineering data displays available; however, host configurations that are required to maintain these displays represent a significant expense to the users. In addition to the increased computing environment costs, are restrictions associated with control of mission data within the security restrictions, i.e., known as the "firewall". These displays meet the minimum requirement for those who reside in the Mission Support Areas (MSAs) behind the firewall; but do not provide enhanced visual or analytical capabilities needed for the growing number of missions expected beyond 2000. Moreover, these displays require UNIX workstations and project specific network connections. Yet, there are a number of scientists and engineers outside of Jet Propulsion Laboratory that require access to monitor their instruments health and status in order to plan their activities. As the number of missions are increasing, while the development and mission operations budgets are decreasing, the need for platform independent display and analysis software allowing remote access is escalating. To meet this demand, the Remote Access Multi-mission Processing and Analysis Ground Environment (RAMPAGE) subsystem is developed.

2.0 RAMPAGE HISTORY

Spacecraft missions at JPL receive engineering data in the telemetry stream containing measurements taken at a specific instance in time. These measurements are referred to as state representations. Mission analysts determine the health of the spacecraft by observing these state representations. In an effort to facilitate mission analysts, RAMPAGE was developed as a tool for acquiring and viewing state representations in near real time.

RAMPAGE was originally developed to test the concept of

[1] The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

web based mission operations for the Galileo mission. The original prototype of RAMPAGE was assembled from several existing Galileo applications. After Galileo's budget was cut drastically, RAMPAGE was ported to the DS1 and Cassini missions.

3.0 RAMPAGE PROTOTYPE ARCHITECTURE

RAMPAGE was initially developed using "n" tier client/server architecture. The main RAMPAGE server reads state representations and their time tags in the telemetry data stream and formats them into predefined tabular and/or plot pages. The reformatted results are stored as ASCII strings in shared memory.

The Tabular Display Builder process accesses shared memory where the state representations and their times are contained, wraps strings with Hyper Text Markup Language (HTML) tags, and writes the result to a disk file which corresponds to a Uniform Resource Locator (URL).

The Plot Display Builder process reads the shared memory segment where a list of state representations and times for states to be plotted are contained, plots these values, and converts these output into a Graphics Interchange Format (GIF) image file. Then, they are embedded in HTML formatted file which correspond to a URL.

The HTML files used client pull to refresh the page at different intervals according to the mission telemetry rate and the type of display plot or tabular. The HTML files were accessible using a Hyper Text Transfer Protocol (HTTP) server[2]. Mission support personnel could access the output of RAMPAGE using freely available web browsers.

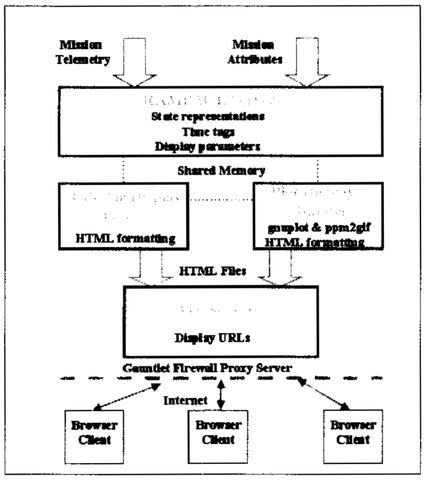

Figure 1 RAMPAGE Prototype Architecture

[2] Apache server for DS1 and Netscape Enterprise server for Cassini.

4.0 DEEP SPACE 1: DEPLOYMENT OF RAMPAGE

In October 1997, the DS1 mission made a commitment to use RAMPAGE for distributed operations rather than requiring remote users (i.e., Primary Investigators) to deploy Advanced Multimission Mission Operations Systems (AMMOS) workstations along with leased dedicated lines. This decision by DS1 placed pressure on RAMPAGE to become completely reliable, which warranted some new features to attain this high degree of availability. As a result, the following capabilities got implemented: (1) constant status updates of the telemetry flow, (2) ability to restart RAMPAGE processes automatically if they were stopped via cron jobs, (3) email notification to specified operators if RAMPAGE was restarted, and (4) creation of HTML forms linked to Common Gateway Interface (CGI) scripts to perform all necessary maintenance on the configuration of RAMPAGE.

Although companies partnering with JPL required local access to data files that were created during DS1 spacecraft telemetry downlink, JPL couldn't afford to provide AMMOS workstations. Thus, the use of RAMPAGE became very attractive to both JPL and remote engineers. In order to meet this need of telemetry data access, HTML forms and corresponding CGI scripts were created so that RAMPAGE was able to query data from the DS1 mission data server which is located behind the firewall. Then, the results of the query were delivered to the query requestor's local computer for further processing.

Initially, RAMPAGE was deployed behind a firewall to meet JPL's security concerns and constraints. However, due to firewall configuration issues such as firewall filtering of all JavaScript pages, the operability of RAMPAGE was greatly hampered.

Lessons Learned

The use of RAMPAGE excited DS1 project users, who were located at JPL as well as remote sites. These users' responses with wide participation from remote sites, RAMPAGE was able to successfully demonstrate its ability of allowing users at any location to participate in accessing mission information by using a simple user interface on any platform. Spacecraft support personnel could stay home on weekends instead of travelling to JPL during the system testing. For instance, analysts at Spectrum Astro were able to track the performance of their Ion Propulsion System from their normal work locations and their homes during weekends.

One of the RAMPAGE success factors stems from seeking out suggestions and feedback from potential users and incorporating them into RAMPAGE development. However, it also became obvious that RAMPAGE had some limitations that needed to be reengineered in order to increase its capabilities to support future JPL missions. These limitations were: (1) the usage of shared memory limited the scalability of RAMPAGE, (2) only 80 tabular displays and 20 plots were supported, (3) the plots consisted of GIF images, which, because of their size, are slow to download over modems, (4) new display types were hard to implement, and (4) firewall issues need to be addressed. To overcome above-mentioned observed limitations, the redesign of RAMPAGE became essential.

5.0 RAMPAGE PRODUCTION SYSTEM

Architecture

In redesigning RAMPAGE, it was recognized that the ability to embrace new technologies without rendering major rewrites of the core software was necessary. Past experience has shown that each project also requires some mission specific capabilities. To provide the flexibility to meet these mission demands, an object-oriented framework was designed. Objects within modules can be replaced with minimal impact on any other modules within RAMPAGE. An additional benefit of this modular approach is load balancing - modules can be assigned to specific processors on a multiprocessor machine or the modules can be run on multiple workstations when the demand becomes too high. In the challenging atmosphere of a Faster Better Cheaper environment at JPL, it becomes almost mandatory to employ an iterative rapid prototyping approach along with concurrent engineering. These approaches also provide opportunities to solicit customer feedback before the final delivery is made. For example, the decision of using either Java Remote Method Invocation (RMI) or Common Object Request Broker Architecture (CORBA) was needed. Based on the following criteria, RMI was selected: (1) RMI presents less overhead than CORBA and (2) all RAMPAGE server applications were written in Java; whereas, CORBA appears to offer more benefits in a mixed language environment.

To address security, RAMPAGE requires all server processes to be deployed behind the JPL Gauntlet firewall with HTTP proxies running on the firewall. To avoid firewall conflicts, remote client access to servers is limited to HTTP protocol and Java RMI is performed only within the firewall. Password authentication is also used to limit external access to each mission's web servers and passwords can be added to any web pages or directories that require additional restrictions. On the client side security is addressed by the fact that Applets from JPL can be trusted. Moreover, the Java Virtual Machine sandbox will provide an additional degree of protection.

IMPLEMENTATION

RAMPAGE still requires files containing mission specific attributes that were inherited from the legacy systems. In the past, these files were kept locally on each mission specific workstation. Now, RAMPAGE has placed all of these mission parameters on a single server. To make these parameters available for processing by RAMPAGE, a SUN UltraSparc 60 server was dedicated to run mySQL database. The "mm" Java Data Base Connectivity (JDBC) driver is used as the Java Application Programming Interface (API).

The mission telemetry rates preclude using the database to store state representations extracted from the telemetry. In order to process the telemetry data efficiently, a cache process was developed whose main purpose was to buffer telemetry data as it flowed from upstream. Therefore, a cache process for each mission was implemented to avoid intermingling of data. If performance becomes an issue, several cache processes can support a single mission. This cache contains only representations of states that are included in a mission display configuration file. The cache consists of lists of measurements and the times corresponding to those measurements. Usually, there are 3 times associated with each measurement: the Spacecraft Event Time (SCET), the Spacecraft clock (SCLK), and the Earth Received Time (ERT). RMI was used as an interface between this process and the other RAMPAGE display applications.

One new display application RAMPAGE has incorporated into this production system is a 3-D graphics. It was intended as a mission planning tool, but was adapted as a spacecraft attitude display tool. This adaptation, which is called FastGraphX, consists of a server providing body position data from JPL Spacecraft Planet Kernel (SPK) and a client that renders graphics.

The FastGraphX server retrieves the attitude data for each mission from the mission cache process, where the server interface to the mission cache process is implemented using Java RMI. This attitude data is passed to the FastGraphX client in the form of a quaternion. The server interface to the client is implemented by using HTTP transfer of specially formatted data files. These data files contain all body positions required by the client to build the display.

The client side of FastGraphX renders the data in a spacecraft centric view and allows the user to display various attributes such as direction vectors to user specified bodies. The user can modify the view by using the mouse or other User Interface elements. Some of the basic features of this display are:

- Accurate attitude representation
- Accurate lighting
- Correct position of other bodies (Sun, Earth, etc.)
- User selectable display content.
- Direction Vectors.

- Predicted versus actual attitude
- Coordinate Axis
- Tabular data (times, distances, etc.)
- Real star map
- Real time modification of the view via mouse or User Interface
- Regular display updates based on telemetry
- Realistic CAD models of spacecraft

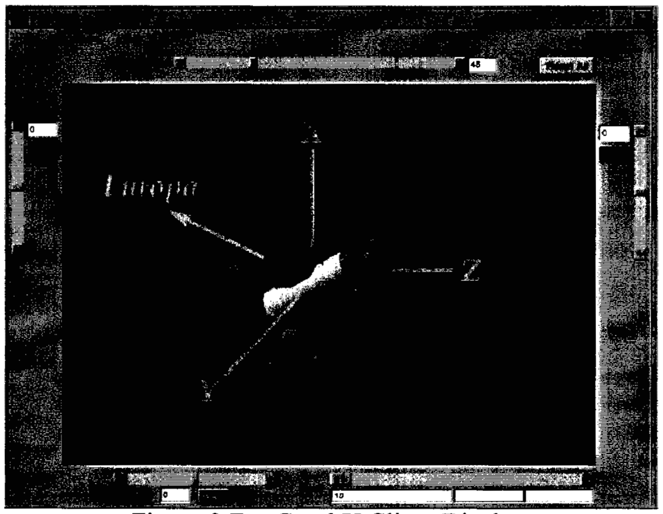

Figure 2 FastGraphX Client Display

The FastGraphX client consists of native compiled binary code and has been ported to SUNs, SGIs, HPs and PCs running Windows NT.

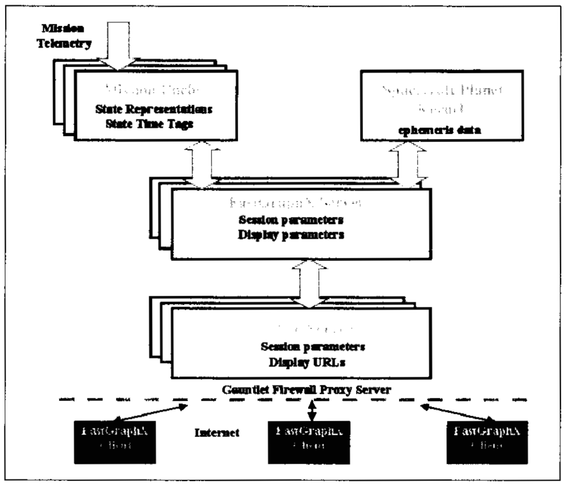

Figure 3 FastGraphX Architecture

A second display application provided by RAMPAGE is the Display and Data Formatter process in response to port old displays that are provided by the legacy systems. Currently, 3 types of displays were identified that RAMPAGE was required to support: tabular displays, plots, and fixed page displays. Fixed page displays are similar to tabular displays, but they provide the capability of additional ASCII fields. A new type of display, a schematic display, is part of the redesign effort to show spacecraft subsystem elements, their relationships and their current state representation.

To reduce development cycle time and cost, a Commercial- Off-The-Shelf (COTS) software package was considered to address the graphics needs for this task . The Generic Logic Toolkit was selected as the graphics builder COTS tool for RAMPAGE. GLG Toolkit met the following criteria set by RAMPAGE:

- GLG Toolkit generates web based cross platform graphics in the form of Java Applets
- These graphics were capable of emulating the legacy displays
- GLG Toolkit is capable of creating more advanced displays such as spacecraft subsystem schematics
- Data required to drive the graphics is ASCII or compressed ASCII instead of GIF files
- Interface to the package is modular, easy to snap in or out

The Display and Data Formatter server process queries the database for display formats and any state specific parameters at startup and uses these parameters to build display templates. The server then uses the mission cache process as a source of state representations. The Display and Data Formatter server process spawns a thread for each remote user request. The thread contains all session data and all display parameters required for building the selected display. At specific intervals, the Display and Data Formatter server queries the mission cache for the state representations that are required to build the display and the client's display is updated by a push from the server.

The client side of the Display and Data Formatter is a web browser that loads an applet selected from a list of predefined display applets. These predefined displays are built using the GLG Toolkit. They can be built by someone with extensive knowledge of the mission or by someone designated to build the displays based on requests from mission personnel.

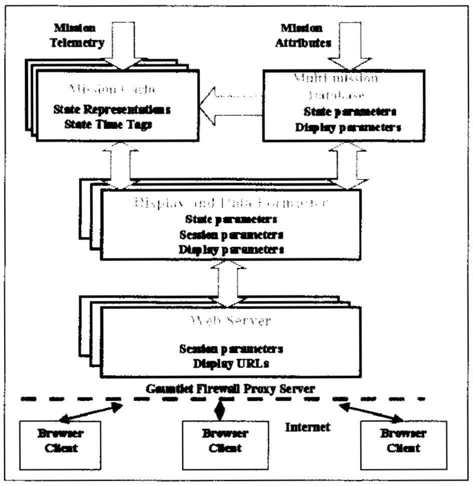

Figure 4 Display and Data Formatter Architecture

To gain wide user support and participation in the testing of RAMPAGE, each mission's URL is advertised to the current and potential users. This approach promotes RAMPAGE acceptance by the users while addressing training needs. In order to obtain a high degree of confidence from the mission management, especially flying missions, new results are verified against the existing systems output through side by side comparison.

6.0 RAMPAGE System Deployment

The first customer of the production version of RAMPAGE will be the Mars01 project. However, Mars01 has added several requirements to RAMPAGE to include web based data management capabilities. The project had already deployed a file management system and wanted to add a web interface to allow remote users' access to their file management system. The current file management system is to be Network File System (NFS) mounted to the mission HTTP server(s). Presently, extensive performance testing is required to determine if this configuration is adequate. One alternative is to run an HTTP server on the file management server host. In addition, the project wanted to use Secure Socket Layer (SSL) capability to enhance their web sites security.

Past experience with other projects had proven the feasibility of using HTTP to move and organize files with certain limitations. If remote users wanted to upload files from their local host to the mission file management system, using HTTP 2 methods were provided.

1. Use a Netscape Version 2.0 or greater browser for manual uploads. This approach requires an HTML form linked to a CGI script that will store the file on the server.
2. Use of libwww library from the World Wide Web consortium to automate uploads. This approach requires the C or PERL library to be installed on the remote machine and a program or script then calls the necessary methods to perform the file upload to the server.

Providing links to the directories containing the mission data files allows downloading of data using a web browser. Since these directories have no index file, the HTTP server is configured to provide Fancy Indexing. This configuration allows remote users to view all of the files in the directory as hypertext links, which can be clicked to transfer the data to the remote user host.

An additional requirement for the Mars01 mission to RAMPAGE is the creation of standing queries through a cron job that request data from the mission data server. Remote users will subscribe to the data they need and provide an email address for notification when a query has been executed. The query is triggered at a specific time and the data is placed in the file management system according to the parameters in the query. Once the query is complete, an email is sent to all users who have subscribed to the query. The email will contain hypertext links to the files created by the standing query. The standing queries must also remove any files over 14 days old to prevent the disk from filling up with outdated files.

The requirement to use SSL resulted in the purchase of the Netscape FastTrack HTTP server. The Apache server could be configured with additional modules to perform SSL but purchase of a product with the SSL capability built in was more desirable. Purchase has been made and its configuration has been implemented. Now, testing has to be performed to verify the effect on system performance.

7.0 Conclusion

Based on lessons learned from supporting multi-mission ground data systems at JPL, the following trends are observed and predicted:

- Analysts will support multiple mission, each mission could be in a different phase (Planning, Assembly, Test, and Launch Operations [ATLO], test, launch, cruise, and/or encounter).
- Analysts will require visualization and analysis tools to utilize historical data.
- Analysts will demand integrated information for correlation and interpretation of state or status of the system.
- Analysts will require easy access to mission data regardless of location and platform.
- Analysts will need ability to analyze data quickly and intuitively.
- Mission requires low cost, low risk and responsive systems.

With those trends in mind, the RAMPAGE system is designed to provide:

- standard interfaces to end-user applications,
- scalability, interoperability, and accessibility,

- platform independence,
- simplified tool and data distribution,
- user customized displays, and
- an integrated system environment for distributed mission operations and science teams.

As a result of RAMPAGE implementation and deployment, several foreseeable benefits are and can be anticipated:

- Hardware cost reduction. (Costly workstations replaced with multi-purpose utilization of Desktop PCs and minimal system administration cost).
- Reduction in project requirement for dedicated landlines and workstations at remote sites.
- Reduction in hardware maintenance cost associated with workstation configurations at local and remote sites.

Young H. Lee is an Advanced Multimission Operations System (AMMOS) Multimission Ground Data System (MGDS) system engineer and technical group supervisor who is currently working Mars 01 project. In the past, she has worked on Voyager, Magellan, Galileo, Mars Observer, Mars Global Surveyor, Mars Climate Orbiter, Mars Polar Lander, Stardust, and Deep Space 1 mission. She has a MSMIS from Claremont Graduate University and currently working on her Doctorate Degree.

Ted Specht has worked on spacecraft ground data systems at JPL for 16 years. During that time, he has worked on the Deep Space Network, Voyager, Ulysses, Galileo, Cassini, Deep Space 1, Mars01 and the Shuttle Radar Topography Missions. He is currently devoted to implementing RAMPAGE for future JPL missions.